

5 DIRECTIVE



5.1 Functions

5.2 If

5.3 Each

5.4 For + While

5.5 Mixin' In



Responsive Refresher

- Straight from Journey Into Mobile:

```
target / context
```

- If the target size of our sidebar is 350px and the context of its parent is 1000px:

```
350px / 1000px = 0.35  
0.35 * 100 = 35%
```

5.1 Functions



application.scss

```
@function fluidize() {  
  @return 35%;  
}  
.sidebar {  
  width: fluidize();  
}
```

always returns 35%

application.css

```
.sidebar {  
  width: 35%;  
}
```

5.1 Functions



application.scss

```
@function fluidize($target, $context) {  
  @return ($target / $context) * 100%;  
}  
.sidebar {  
  width: fluidize(350px, 1000px);  
}
```

application.css

```
.sidebar {  
  width: 35%;  
}
```

5.1 Functions



- More on responsive design + Sass later, including a built-in `fluidize` replacement
- Function arguments = **same rules** as mixin arguments

5.1 Functions



Using `@if`, we can conditionally output code:

application.scss

```
$theme: dark;

header {
  @if $theme == dark {
    background: #000;
  }
}
```

application.css

```
header {
  background: #000;
}
```

5.2 If



Comparisons

- ⦿ == equal to
- ⦿ != not equal to
- ⦿ > greater than *
- ⦿ >= greater than or equal to *
- ⦿ < less than *
- ⦿ <= less than or equal to *

* numbers only

5.2 If



application.scss

▶ \$theme: light;

```
header {  
  @if $theme == dark {  
    background: #000;  
  }  
}
```

5.2 If



@else provides a fallback if everything evaluates false or null:

application.scss

```
$theme: light;

header {
  @if $theme == dark {
    background: #000;
  } @else {
    background: #fff;
  }
}
```

application.css

```
header {
  background: #fff;
}
```

5.2 If



@else if allows for multiple comparisons:

application.scss

```
$theme: pink;

header {
  @if $theme == dark {
    background: #000;
  } @else if $theme == pink {
    background: pink;
  } @else {
    background: #fff;
  }
}
```

application.css

```
header {
  background: pink;
}
```

5.2 If



5.1 Functions



5.2 If



5.3 Each



5.4 For + While



5.5 Mixin' In



Iterating Over a List

- The `@each` directive allows us to loop through each list item:

```
$authors: nick aimee dan drew;
```



application.scss

```
$authors: nick aimee dan drew;

@each $author in $authors {
  .author-#{ $author } {
    background: url(author-
#{ $author }.jpg);
  }
}
```

\$author →

- 1: nick
- 2: aimee
- 3: dan
- 4: drew

application.css

```
.author-nick {
  background: url(author-nick.jpg);
}
.author-aimee {
  background: url(author-aimee.jpg);
}
.author-dan {
  background: url(author-dan.jpg);
}
.author-drew {
  background: url(author-drew.jpg);
}
```

5.3 Each



```
.item {  
  position: absolute;  
  right: 0;  
  @for $i from 1 through 3 {  
    &.item-#{ $i } {  
      top: $i * 30px;  
    }  
  }  
}
```

$\$i \longrightarrow$

1:	1
2:	2
3:	3

```
.item {  
  position: absolute;  
  right: 0;  
}  
.item.item-1 {  
  top: 30px;  
}  
.item.item-2 {  
  top: 60px;  
}  
.item.item-3 {  
  top: 90px;  
}
```

5.4 For + While



- `@for` and `@while` = `@each` with more control
- `@while` requires manually updating the index

5.4 For + While



application.scss

```
$i: 1;

.item {
  position: absolute;
  right: 0;
  @while $i < 4 {
    &.item-#{ $i } {
      top: $i * 30px;
    }
    $i: $i + 1;
  }
}
```

$i \rightarrow$


1:	1
2:	2
3:	3
4:	4

application.css

```
.item {
  position: absolute;
  right: 0;
}
.item.item-1 {
  top: 30px;
}
.item.item-2 {
  top: 60px;
}
.item.item-3 {
  top: 90px;
}
```

5.4 For + While



 `$i: 2;` `.item {
 position: absolute;
 right: 0;
 @while $i <= 6 {
 &.item-#{ $i } {
 top: $i * 30px;
 }
 $i: $i + 2;
 }
}``$i` `1: 2``2: 4``3: 6``4: 8`

```
.item {  
  position: absolute;  
  right: 0;  
}  
.item.item-2 {  
  top: 60px;  
}  
.item.item-4 {  
  top: 120px;  
}  
.item.item-6 {  
  top: 180px;  
}
```

5.4 For + While



5.1 Functions



5.2 If



5.3 Each



5.4 For + While



5.5 Mixin' In



Mixins

- *Similar* sets of properties used multiple times with small variations

Extend

- Sets of properties that match *exactly*

Functions

- Commonly-used operations to determine *values*



_buttons.scss

```
@mixin button($color, $rounded: true) {  
  color: $color;  
  @if $rounded == true {  
    border-radius: 4px;  
  }  
}  
.btn-a {  
  @include button(#000, false);  
}  
.btn-b {  
  @include button(#333);  
}
```

application.css

```
.btn-a {  
  color: black;  
}  
.btn-b {  
  color: #333333;  
  border-radius: 4px;  
}
```

5.5 Mixin' In



```
@mixin button($color, $rounded: false) {  
  color: $color;  
  @if $rounded {  
    border-radius: $rounded;  
  }  
}  
.btn-a {  
  @include button(#000);  
}  
.btn-b {  
  @include button(#333, 4px);  
}
```

used if \$rounded
isn't false or null

```
.btn-a {  
  color: black;  
}  
.btn-b {  
  color: #333333;  
  border-radius: 4px;  
}
```

5.5 Mixin' In



ASSEMBLING SASS



PROPERTY OF:
CODE SCHOOL

PROJECT DATE:
SEP, 2012



This project was created and developed by
David E. Johnson, M.Ed., S. M.Ed.
and is the property of the
Assembly School. It is not to be
reproduced or distributed in any form
without the written consent of the
Assembly School. It is the property of
David E. Johnson.

S

STANDARD
INSPECTION : NO 18